MATLAB to iPhone Demo

Last Update: September 15, 2015

Tested with

- MATLAB R2015b
- MacBook Pro 15” (2013), OS X 10.10 Yosemite, XCode 7 beta
- iPhone 5S with iOS 8.4

Goal

This demo is to show the workflow of going from a MATLAB algorithm, generating C code, import into an existing XCode project, compile & download & run the app on your iPhone or iPad.

Preparation

1. Hardware:
    1. Mac or MacBook, preferably 10.10 (Yosemite) or newer
    2. iPhone 5S
2. Software
    1. XCode, preferably 7
    2. MATLAB R2015b
3. Set up Xcode if needed
    1. Start XCode, and if you’re using Xcode for the first time, this will ask you to select a directory for Xcode projects
    2. Download and open GLGravity project
4. Test if you can build the project for the simulator
    1. Choose to build for either the iPhone or iPad simulator
    2. Press the “play” button to build, download, and run the simulator
5. Test if you can build the project for the iPhone

Run the demo

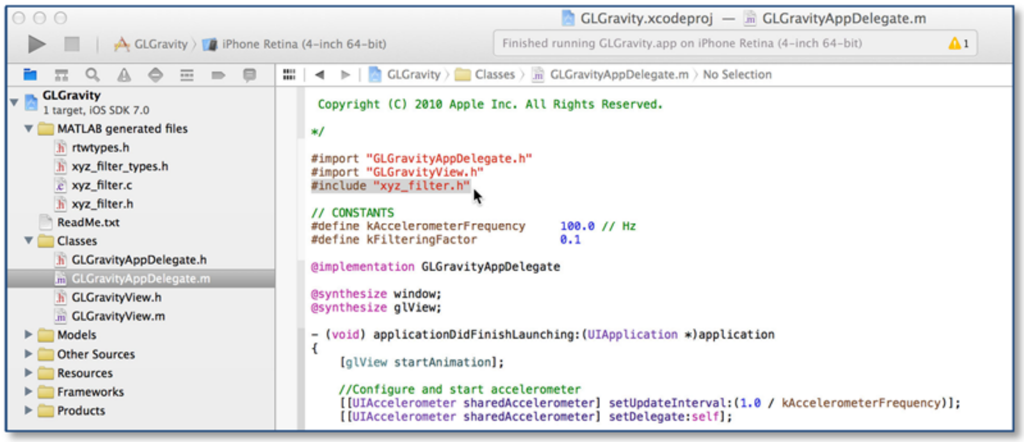
1. Generate C code for xyz_filter.m

   1. Execute run_demo.m to start the workflow
   2. You can generate code graphically through the MATLAB Coder Project window (coder xyz_filter.prj) or through the command line (make_xyz_filter.m)
   3. At the end of the day, you should have these files:
      1. xyz_filter.c
      2. xyz_filter.h
      3. xyz_filter_types.h
      4. rtwtypes.h
2. Integrate generated code into GLGravity project
   1. Open Xcode and then the GLGravity project
   2. Create a new group called "MATLAB generated files"
   3. Drag and drop the 4 generated files (mentioned in step 1.3 above) into the group
      1. Choose to link the files (and not copy the files); this way, whenever you generate code again in MATLAB, Xcode will be able to get the newest files
   4. Modify GLGravityAppDelegate.m in GLGravity\Classes so it will call the generated filter that we just generated
      1. See the MATLAB to iPhone and Android Made Easy webinar for step-by-step instructions (in particular starting around 20:08)
      2. See Integrate MATLAB generated C code into GLGravity project from scratch below for text instructions
   5. Choose the simulator, build, and run it to confirm everything works
   6. Choose the physical iPhone, build, and show it in action
3. To "break" or show that the generated code is actually being executed in the simulator or real iPhone, you can change the cutoff frequency to 0.1 Hz
   1. Go back to MATLAB and open create_xyz_filter_for_codegen.m
   2. Find this line
      1. Fc = 2.5;   % Cutoff frequency (Hz);
   3. Change Fc to 0.1
   4. Regenerate code and show the results. You should see the teapot move very slowly when you move around

Integrate MATLAB generated C code into GLGravity project from scratch

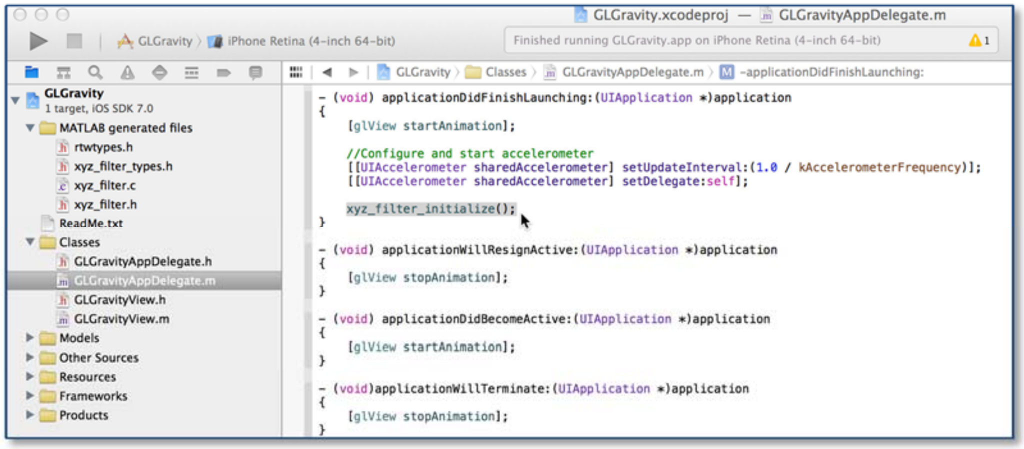
If you need to integrate code from a fresh copy of the original GLGravity.zip, here’re the steps to do it:

1. Start at the point after you’ve generated the 4 files from the MATLAB project
2. Download, unpack, and open the fresh GLGravity project
3. There is main.m under “Other Sources”, but it’s not something we touch. Instead, we need to modify “GLGravityAppDelegate.m” under “Classes”
4. Import the MATLAB generated files. Under the main project “GLGravity”, right-click and select “New Group”
5. Name the new group “MATLAB generated files”
6. Open Finder, navigate to the directory of the MATLAB generated files. Should be under “\codegen\lib\xyz_filter\”
7. Select the four .c and .h files, and drag & drop into GLGravity project, best under the newly created “MATLAB generated files” group
8. Xcode will ask for option for adding these files. Keep everything with default options (in particular, do **NOT** check the “Copy items into destination group’s folder). The Xcode project will keep a reference to the files in that directory; hence, in subsequent MATLAB code generation sessions, you do not need to keep copying new files to the Xcode project (since these files are referenced in the Xcode project)
9. Open GLGravityAppDelegate.m and make 4 changes:
    1. Include xyz_filter.h as a #include statement (i.e. #include “xyz_filter.h”)



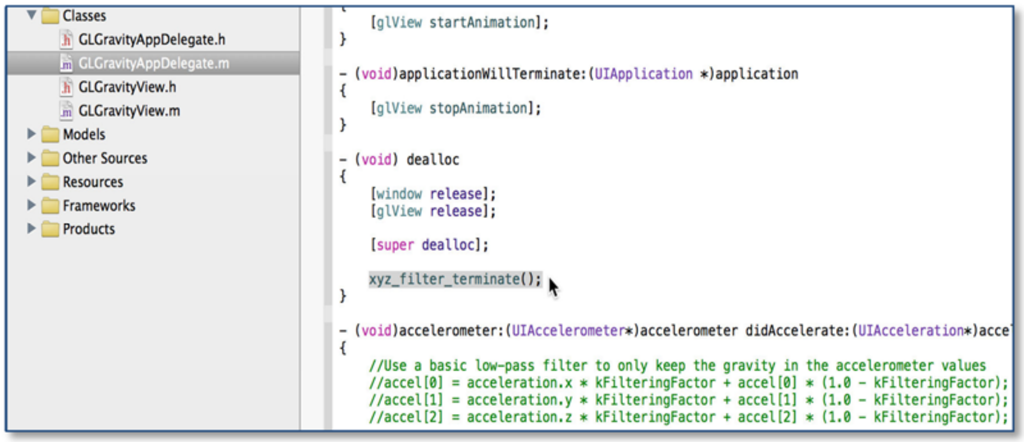
```
 Copyright (C) 2010 Apple Inc. All Rights Reserved.

*/

#import "GLGravityAppDelegate.h"
#import "GLGravityView.h"
#include "xyz_filter.h"

// CONSTANTS
#define kAccelerometerFrequency     100.0 // Hz
#define kFilteringFactor            0.1

@implementation GLGravityAppDelegate

@synthesize window;
@synthesize glView;

- (void) applicationDidFinishLaunching:(UIApplication *)application
{
    [glView startAnimation];

    //Configure and start accelerometer
    [[UIAccelerometer sharedAccelerometer] setUpdateInterval:(1.0 / kAccelerometerFrequency)];
    [[UIAccelerometer sharedAccelerometer] setDelegate:self];
```

2. Initialize xyz_filter. Under applicationDidFinishLaunching:(UIApplication *)application, add “xyz_filter_initialize();”



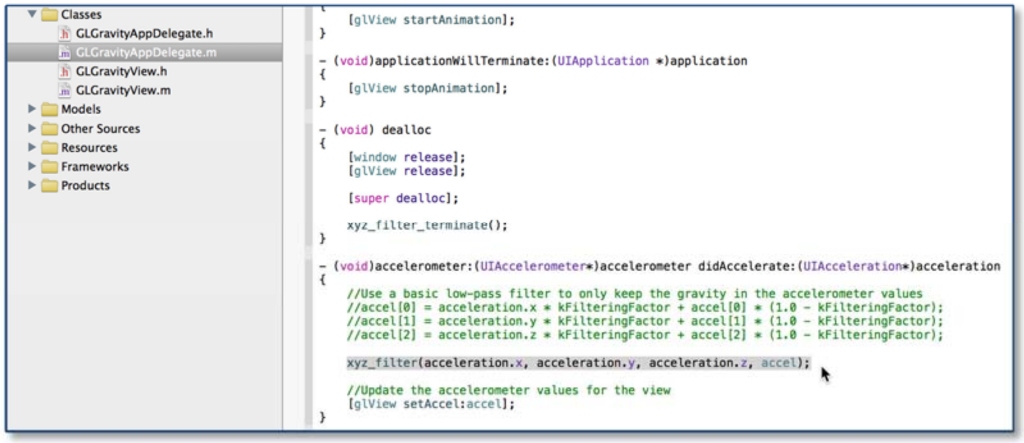
```
- (void) applicationDidFinishLaunching:(UIApplication *)application
{
    [glView startAnimation];

    //Configure and start accelerometer
    [[UIAccelerometer sharedAccelerometer] setUpdateInterval:(1.0 / kAccelerometerFrequency)];
    [[UIAccelerometer sharedAccelerometer] setDelegate:self];

    xyz_filter_initialize();
}

- (void) applicationWillResignActive:(UIApplication *)application
{
    [glView stopAnimation];
}

- (void) applicationDidBecomeActive:(UIApplication *)application
{
    [glView startAnimation];
}

- (void)applicationWillTerminate:(UIApplication *)application
{
    [glView stopAnimation];
}
```

3. Deallocate xyz_filter after the app closes. Under dealloc, add “xyz_filter_terminate();”



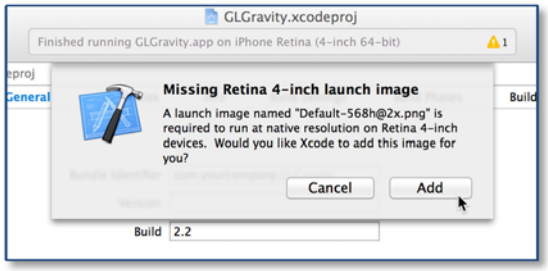
```
    [glView startAnimation];
}

- (void)applicationWillTerminate:(UIApplication *)application
{
    [glView stopAnimation];
}

- (void) dealloc
{
    [window release];
    [glView release];

    [super dealloc];

    xyz_filter_terminate();
}

- (void)accelerometer:(UIAccelerometer*)accelerometer didAccelerate:(UIAcceleration*)acce
{
    //Use a basic low-pass filter to only keep the gravity in the accelerometer values
    //accel[0] = acceleration.x * kFilteringFactor + accel[0] * (1.0 - kFilteringFactor);
    //accel[1] = acceleration.y * kFilteringFactor + accel[1] * (1.0 - kFilteringFactor);
    //accel[2] = acceleration.z * kFilteringFactor + accel[2] * (1.0 - kFilteringFactor);
```

4. Insert the new acceleration parameters. Under didAccelerate:(UIAcceleration*)acceleration, add xyz_filter(acceleration.x, acceleration.y, acceleration.z, accel);
    1. Make sure to remove references to the old acceleration calculation, i.e. accel[0] = …, accel[1] = …, accel[2] = … by commenting them out



```
    [glView startAnimation];
}

- (void)applicationWillTerminate:(UIApplication *)application
{
    [glView stopAnimation];
}

- (void) dealloc
{
    [window release];
    [glView release];

    [super dealloc];

    xyz_filter_terminate();
}

- (void)accelerometer:(UIAccelerometer*)accelerometer didAccelerate:(UIAcceleration*)acceleration
{
    //Use a basic low-pass filter to only keep the gravity in the accelerometer values
    //accel[0] = acceleration.x * kFilteringFactor + accel[0] * (1.0 - kFilteringFactor);
    //accel[1] = acceleration.y * kFilteringFactor + accel[1] * (1.0 - kFilteringFactor);
    //accel[2] = acceleration.z * kFilteringFactor + accel[2] * (1.0 - kFilteringFactor);

    xyz_filter(acceleration.x, acceleration.y, acceleration.z, accel);

    //Update the accelerometer values for the view
    [glView setAccel:accel];
}
```

10. Build and download to target. Everything should work correctly
11. Initially, you may have an error messaging saying that the application does not have an image for 4” retina screens (and if you launch the app on your 4” iPhone, it will not work).
    1. To fix, select the “let Xcode find this file” option, and you should fix this issue


GLGravity.xcodeproj

Finished running GLGravity.app on iPhone Retina (4-inch 64-bit)

Missing Retina 4-inch launch image

A launch image named "Default-568h@2x.png" is required to run at native resolution on Retina 4-inch devices. Would you like Xcode to add this image for you?

Build: 2.2